87221

Experimental Physics and Industrial Control System (EPICS) Application Source/Release Control for EPICS R3.11.6

Bob Zieman and Marty Kraimer

March 25, 1994

CHAPTER 1 Introduction

1.1 Overview

This manual describes a set of tools that can be used to develop software for EPICS based control systems. It provides the following features:

Multiple Applications

The entire system is composed of an arbitrary number of applications.

Source/Release Control

All files created or modified by the application developers can be put under sccs, which is a Unix Source/Release control utility.

Multiple Developers

It allows a number of application developers to work separately during the development phase but combine their applications for system testing and for a production system.

Makefiles

Makefiles are provided to automatically rebuild various application components. For C and state notation programs, Imakefiles are provided.

1.2 Definitions

Application System Area

The set of directories and files managed by the tools described in this document. Everything is stored in one directory tree. The top level directory contains information common to all applications in this area as well as a subtree for each application.

Application

A subtree under the application system area that contains all the files for a single application.

Application Shadow Area

A set of directories and soft links to an application system area. It appears to the user just like a copy of the system area.

Application Production Area

A copy of a working application system area for use by operations.

1.3 Classes of Users

Application System Manager

The Application System Manager is responsible for the Application System area.

Application Developer

Anyone who tests, modifies, or extends an application's software. If multiple developers are working on the same system each should develop in a private shadow area.

Application Production Manager

A person responsible for production application software.

1.4 Document Conventions and Information

The following conventions and/or representations apply to the remainder of this document.

<epics>	Represents the full path name of an EPICS release
<top>	Represents the root node of an application system area. It is the directory from which we can access EPICS components.
<shadow>	Represents the top node of an application shadow node. "shadow node" and "shadow area" are synonymous.
<archV>	Represents the vxWorks target architecture. Currently hkv2f for the 68020 and mv167 for the 68040.
<archU>	Represents the Unix architecture. Currently sun4.
%	Indicates a prompt for user input or activity.
<EDIT>	Means: edit the file according to the sccs rules. Refer to the sccs procedures in section 1.6. Note: **<EDIT>** includes doing a delget if in the system area.

1.5 Environment

In order to use the EPICS Unix tools in designated directories you must add the following line to your .cshrc file:

set path = ($path ./.epicsUnix/'arch'/bin)

1.6 Source/Release control commands

The Unix sccs utility is used to put all user editable files under source/release control. The unix documentation should be consulted for a complete description of sccs. This section gives a brief description of the commands normally used by application developers. Wherever **<filename>** is shown a list of filenames is allowed.

%sccs create <filename>

This command places a file under sccs control for the first time. After the file is placed under sccs control a read only copy is created, i.e. an sccs edit command must be issued before the file can be modified.

This command also creates a backup copy of the original file. It is the original file with a comma prepended to the name. It is a good idea to remove this file.

%sccs edit <filename>

This command checks out a file so that it can be modified. If a file is checked out in an application shadow area, other developers will not see any modifications until an sccs delta is executed.

If a developer checks out a file in a shadow area it is actually checked out from the system area, i.e. no other developer can try to modify the same file. Other developers do not, however, see any changes made in the shadow area until the developer checks in modified files.

%sccs unedit <filename>

This command causes the SCCS directory to revert to the state it was in before the last sccs edit <filename> command was issued.

%sccs delta <filename>

This command checks in a modified file. This should only be done when everyone attached to an application system area is expected to see the changes. A new sccs version of the file is created. It is possible to retrieve previous versions.

It is also possible to issue an "sccs delget", which combines an sccs delta with an sccs get, and an "sccs deledit", which combines an sccs delta with an sccs edit.

%sccs get <filename>

This command retrieves a read only version of the file. This is done automatically by one of the tools described in this manual. It is also useful in a shadow area when the user wants to make a temporary change to a file, e.g. for debugging purposes. In this case the user must change the file protections before it can be modified.

%sccs info

This command displays a list of all checked out files in the directory from which the command is issued.

1.7 Getting started

The normal procedure for getting started is to:

Create a system area

This is done by the application system manager. See sections 3.1 – 3.3 for details.

Get Application Specific ascii definition files.

All application developers using this application system area must agree on a common set of ascii definition files. See section 5.3 for instructions.

After all ascii files are installed run makesdr in <top>.

Populate Each Application

The developers for each application should install all files related to each application. This includes Unix sources, IOC sources, and databases. Chapter 5 explains how to install each component. It is up to the application system manager and the application developers to decide if it is easier to do this in the system area or if each application developer should do his/her part in a shadow area.

Prepare each IOC for booting

Modify the startup files in each ioc directory.

Perform the normal Integration steps

Perform the steps given in section 3.4.

At this time you should have a working application system area.

CHAPTER 2 Application System Area Architecture

The root directory of the Application System Area and its contents is referred to as <top>. Appendix A shows the file structure stored under <top>

2.1 <top>

2.1.1 User Editable Files

applList	List of all applications.
iocList	List of all iocs.

2.1.2 EPICS related links and files

The following are soft links to epics directories or files.

makefile@	
ascii@	ascii definition files
epicsH@	include files
share@	various epics shared sources
Unix@	directory for locating unix tools
config@	directory containing files needed by source/release tools
vxWorks<archV>@	vxWorks boot image
target<archV>@	directory containing EPICS vxWorks executables
vw@	The location of vxWorks components

The following files identifies EPICS releases.

current_rel	The current release
.current_rel_hist	A history of all getrel commands for this system

The following are for record definitions. The default files are initially links to EPICS and sdrH does not exist. Makesdr creates these if they are missing or out–of–date.

default.dctsdr	
default.sdrSum	
sdrH/rec/	The include files for each record type.

2.2 <top>/cat_ascii

This is the place to store ascii definition files to be added to the end of EPICS definition files.

2.3 <top>/replace_ascii

This contains ascii definition files that are not part of EPICS and also files that replace EPICS supplied files. It is also the place to store C include files containing definitions used in ascii definition files.

2.4 <top>/ioc/<iocName>/

These directories (one for each IOC) contain soft links to the EPICS components needed to boot an IOC. Each also contains a st.cmd<archV> file which must be customized for the particular ioc. Any modules to be loaded into the ioc must be referenced from the st.cmd<archV> file in a relative fashion. If other ioc specific files are needed this is the place to put them. All user created files should be placed under sccs control.

2.5 <top>/<app>App/ – Root of an Application

Root node of an application.

2.5.1 <top>/<app>App/src/

This directory contains files that are meant to be edited by the application developer. This includes the following:

C source.
Include files
State sequence programs.
ImakefileVx and ImakefileUnix.

2.5.2 <top>/<app>App/archList

This file initially contains entries for each target architecture directory to be built. (ex: sun4, mv167, hkv2f). You may remove the entries you do not want built.

2.5.3 <top>/<app>App/<archV>/

This is the directory for building application specific IOC components. It contains a link to <src>/ImakefileVx so that Buildit and make can be executed.

2.5.4 <top>/<app>App/<archU>/

This is the directory for building application specific Unix components. It contains a link to <src>/ImakefileUnix so that Buildit and make can be executed.

2.5.5 <top>/<app>App/<*>Db/

Each application can have an arbitrary set of database directories and each database directory can contain an arbitrary number of IOC databases.

2.5.6 <top>/<appName>/op/

This directory initially contains the following subdirectories:

adl/
alh
ar/.
burt
km/

These directories contain files, for the specified tools, that the application developer wants to place under sccs control. The user may also define other directories and contents in this area.

CHAPTER 3 Procedures for Application System Area

This chapter describes procedures that can only be executed in the system area not in a shadow area. In general any procedure that creates a new directory must be executed in the system area. This includes:

new application directories
new ioc directories
new database directories within an application

It is up to the application developer to notify the application system manager of any new structures to be introduced.

Any **<EDIT>** commands shown in this chapter could also be executed in a shadow area but it is often easier to perform them in the system area.

3.1 Creating The Initial Application System Area

Setup the <top> Directory.

%mkdir <top>
%cd <top>
%<epics>/Unix/share/bin/getrel <epics>
The getrel command to be executed must be executed from the same EPICS release that you will be using.
%apCreateTop
This command creates any missing <top> directory components of an application system area.
%apFixLinks

3.2 Creating Application Specific Directories

3.2.1 Creating New Applications

In order to create the initial applications or add new applications execute the following commands:

%cd <top>
% <EDIT> appList
If appList is not out for edit issue the command "sccs edit appList". Use your favorite editor to add one or more application names to the file. For example:
appName1
appName2
%apCreateApp
%cd <top>/<appName>
% <EDIT> archList
Use your favorite editor to remove unwanted target architecture names from the file.

3.2.2 Creating New Database directories

%cd <top>/<app>App/
%apCreateDbDir

This tool is interactive. You are prompted for each database name, followed by a prompt for the database editing tool (gdct or dct) to be used in that directory.

Note that only one database editor (either gdct or dct) may be used in a given directory.

3.3 Creating IOC directories

%cd <top>
%<EDIT> iocList

Use your favorite editor to add the new ioc names to the file.
example:

iocName1
iocName2

%apCreateIocName

For each name in iocList, this tool creates a **<top>/ioc/<iocName>** directory and populates it with "default" templates and links. Each new directory contains an IOC startup file. The user should modify each st.cmd<archV> file to include the application components and/or instructions to load and run the application.

3.4 Integration

Integration means going to a new release of EPICS or updating the application system area to reflect changes made by application developers in their shadow nodes.

The primary steps are as follows:

1 The application system manager coordinates with the application developers.

2 The application developers should make sure that all DCT databases have an up to date short form report (.rpt files) which is not out for sccs edit.

3 The application developers check in (sccs delta) from their shadow nodes any files they want to be part of the new system.

4 If a new EPICS release is desired, the application system manager issues the following commands:

%cd <top>
%<epics>/Unix/share/bin/getrel <epics>

where <epics> is the full path name to the new version of EPICS.

%apCreateTop
%apCreateApp
%apCreateIocName
%apFixLinks
%make doFixRptDct Note: DCT rpt files must NOT be out for sccs edit. make doFixRpt may be run instead of make doFixRptDct if make doFixRptDct has been run once in the application system area.

5 The application system area is rebuilt by the command:

%make world

See section 6.4.1 for what this command does.

6 After the application system area has rebuilt successfully, application developers can resync their shadow nodes as described in 4.2.

CHAPTER 4 Shadow Node Procedures

This chapter describes procedures that apply only to an application shadow node. An application shadow node is an image of a complete application system area. When created it contains soft links to files in the application system area. The application developer should perform all development in a shadow are rather than the system area so that other developers do not see his/her changes until sccs deltas are executed.

4.1 Creating an application shadow node.

%mkdir <shadow>

NOTE: Do not do this in an application system area.

%cd <shadow>

%<epics>/Unix/<archU>/bin/apCreateShadow <top>

You will be asked to create a file with the touch command.

%touch .applShadow

%<epics>/Unix/<archU>/bin/apCreateShadow <top>

An application shadow area is identical to an application system area with the following exceptions:

1 All files in an application shadow node are initially links.

2 All directories in an application shadow node are real except for each SCCS directory which is a symbolic link.

3 DCT Db directories only have the <*>.database file as a link initially.

4.2 Synchronizing an application shadow node.

Any time the application system area is rebuilt or changed the application developer must synchronize his or her application shadow node.

%cd <shadow>

%apStatusSync

Repeat this step until the status is correct.

%<edit apRemove Script via your favorite editor>

%apRemoveScript

This removes the out–of–date files and directories.

%apCreateShadow <top>

The **apStatusSync** tool is designed to be invoked one or more times before producing the correct **apRemoveScript** file. Status reports go to standard out and commands to remove shadow area components are placed into the **apRemoveScript** file. **apStatusSync** should be repeated until it runs successfully. It is the application developer's responsibility to determine when the status is correct. **apRemoveScript** contains a set of (commented out) unix commands to remove obsolete or illegal files and/or directory components in the application shadow area. It is the application developer's responsibility to edit the **apRemoveScript** file.

CHAPTER 5 Adding/Modifying Components

This chapter describes procedures for adding or modifying application components. These procedures will work in the either the application system area or in a shadow area. If they are issued in the system area remember that all users may be affected. Wherever this chapter refers to **<shadow>** it is also possible to use **<top>**.

5.1 IOC Databases

The procedures given in this section assume that the user has changed to a database directory, i.e. one of the following commands has been issued:

%cd <shadow>/<applName>/<db>Db

or

%cd <top>/<applName>/<db>Db

5.1.1 DCT databases

DCT is used to create new databases and/or modify existing databases. This subsection describes application source/release tools that allow the .rpt files to be placed under source/release control.

DCT causes a problem for source/release control because DCT generates many files for a single database. What is put under sccs control is the short form report file, which must have a file extension of .rpt.

For each database in a database directory the following files can exist:

<file>.rpt	This is the file that is placed under source/release control.
<file>.rpt0	This file is generated by the make utility described below. If such files appear after running make, the user should resolve the differences and make sure that only <file>.rpt remains.
<file>.rpt1, etc	If the user runs make without resolving differences then make keeps creating new files.
<file>.rpt.err	After checking for real errors these can be deleted.
<file>Db.database	This is the file generated by DCT. The make utility described below automatically adds "Db" to the report file name.
<file>Db.ai, etc	The other files generated by DCT.

Report files and source/release control

All report files should be managed via the dbsccs commands described in section 5.1.3.

Update databases from .rpt files that are under sccs control

%make

For each <file>Db.database file with a report file that is under sccs control, make performs the following steps when the <file>Db.database is out of date with respect to the <file>.rpt, default.dctsdr, or default.sdrsum files.

a If <file>Db.database does not agree with it's associated <file>.rpt then a new short form report <file>.rpt0 is generated (NOTE: If this file already exists it uses <file>.rpt1, etc). In this case a warning message is also issued.

b It deletes existing <file>Db.* files and uses atdb to create a read only <file>Db.database file from the <file>.rpt.

Note that the new .database file agrees with the original .rpt NOT the .rpt0 file.

5.1.2 GDCT databases

Refer to the GDCT User's Manual for details. For each database the following files exist:

<file>	The file containing graphical information.
<file>.db	A loadable ascii file used by GDCT.

All new <file> and <file>.db files must be placed under source release control via sccs create commands. The sccs edit, sccs get, and sccs delta commands may be issued as necessary.

It is recommended that GDCT databases are loaded with the dbLoadRecords and dbLoadTemplate commands rather than dbLoad.

5.1.3 dbsccs for DCT databases

Each database is normally represented as a single link to the <dbname>Db.database. This eliminates a lot of clutter in the DCT database directory. The dbsccs tool allows an application developer to edit specific databases.

DCT should be used to create a new <dbname>Db database, and dbsccs create should be used to create the <dbname>.rpt file and put it under sccs control. Once the <dbname>.rpt is under sccs control, DCT can be used to modify the <dbname>Db database only when the <dbname>.rpt file is out for edit.

%dbsccs create <dbname>.rpt

This command performs the following functions:

If <dbname>.rpt is already under sccs control the command aborts.
If <dbname>Db.database is missing, a link or not writable the command aborts.
If a <dbname>.rpt file already exists, it is renamed <dbname>.rpt0. (NOTE: If this file already exists it uses <dbname>.rpt1, etc)
dbta is invoked to create a <dbname>.rpt file.
sccs create is invoked on the new <dbname>.rpt file. The writable <dbname>.Db database files are removed.
atdb is used to create a read only <dbname>.Db database file from <dbname>.rpt.

%dbsccs edit <dbname>.rpt

This command performs the following functions:

If <dbname>Db.database is writable the command aborts.
sccs edit is used to take <dbname>.rpt out for edit.
The <dbname>Db.* files are removed.
DCT is invoked to create a writable <dbname>Db database.

NOTE: If a writable <dbname>Db database exists, DCT can be used to delete the database before issuing dbsccs edit.

%dbsccs delget <dbname>.rpt

This command performs the following functions:

If <dbname>.rpt is missing, a link or not writable dbsccs aborts
dbta is invoked to create <dbname>.rpt file from the <dbname>Db database.
The writable DCT <dbname>Db database files are removed.
atdb is invoked to create a read only <dbname>Db.database file.
sccs delget is invoked for <dbname>.rpt

%dbsccs unedit <dbname>.rpt

This command performs the following functions:

sccs unedit is invoked on <dbname>.rpt
A backup rpt is made if <dbname>.Db.database was modified.
The writable <dbname>.Db database files are removed.
atdb is used to create a read only <dbname>.Db database file from <dbname>.rpt.

5.2 IOC Configuration Files

For each ioc an ioc directory exists under <top>/ioc. In each such directory a vxWorks startup file exists for each supported vxWorks board support package. In addition resource files can also be placed under source/release control.

5.2.1 vxWorks startup files

The startup files (for example <top>/ioc/<iocname>/st.cmd<archV>) must be modified after initial creation and when the set of databases to be loaded changes. The normal sccs edit and sccs delta commands should be used as necessary. When initially created the startup files are prototypes, which contain modification instructions.

5.2.2 resource.def files

The iocInit command in the startup file can have an optional "resource.def" parameter. If it does, then the resource.def file is processed. This file should appear in the same directory as the startup file. It should be placed under source/release control with the sccs create command. Commands sccs edit and sccs delta can be used as necessary.

5.3 Ascii definition files

All applications under <top> must share the same set of ascii definition files. Two directories are available for application ascii files. <top>/replace_ascii/ is the place to store files that are replacements for EPICS files and <top>/cat_ascii/ is the place to store files that don't exist in EPICS and files that should be added to the end of EPICS files.

The command makesdr must be run any time there is a change to any ascii input file used by makesdr. After makesdr completes all databases must be rebuilt and any affected record or device support must be rebuilt.

After makesdr is executed all applications must be rebuilt. The following commands will rebuild all applications:

%cd <top>
%make

This will rebuild all out–of–date applications.

If, however, you are working in a shadow area and are only dependent on a single application, the following commands can be used to rebuild the single application:

%cd <shadow>
%makesdr
%cd <app>App
%make

5.4 Adding Application Specific Source files

All application specific source files are put in <top>/<app>App/src. When any new file is placed in this directory it should be put under source/release control via the sccs create command. Depending on the type of file other files will have to be edited.

Unix Source Files

%cd <src>
%<EDIT>ImakefileUnix

Edit this file to build the new Unix component.

%cd <app>App/<archU>
%Buildit
%make

IOC Source files

This includes C sources and sequence programs. If the sources are for record, device, or driver support remember that ascii definition files must be prepared and makesdr executed as described above. In addition the following must be performed:

%cd <src>
%<EDIT>ImakefileVx

Edit this file to build the new IOC component.

%cd <app>App/<archV>
%Buildit
%make

5.5 Modifying Existing Application Sources

In this case just edit the source in <src> and then execute the make command in either the <archU> or <archV> directory.

5.6 Operator Files

All files placed in the op directory should be managed via the sccs commands.

5.7 Building a single Application

%cd <shadow>/<appName>/

%make

This does a make in each defined <archV>, <archU> and also each <*>Db directory. If a "makefile.pvt" makefile exists it is then invoked.

Individual application components can be rebuilt by qualifying the make command:

%make doGets

This brings all sccs files in this directory and below up–to–date.

%make bldDb

This recreates each database as described in section 5.1.

%make bldMakefiles

This performs a Buildit and make in each defined <archV> and <archU> directory.

%make bldPvt

If a file named makefile.pvt exists then a make is performed using this file.

All of the above can be performed by issuing the command:

%make world

CHAPTER 6 Source/Release Tools

6.1 Application System Area

This section describes tools that should be issued only in an application system area, NOT in a shadow area.

6.1.1 Tools invoked in the <top> directory

getrel

This command is executed in <top> to get a new release of EPICS. It is always issued in <top>. When issued it must be executed with a full path name to the release of epics desired.

apCreateTop

This command must always be issued in directory <top>. The first time this command is issued, it creates all directories and files needed for an application system area. It must also issued whenever a new release of epics is obtained via the getrel command. In this case it makes sure that application system area is correct for the new release.

apCreateApp

This command creates the directories needed for each application that resides under <top>. It is executed in <top> whenever new applications are added to file <top>/appList.

apCreateIocName

This command creates the directories needed for each ioc that resides under <top>/ioc. It is executed in <top> whenever new iocs are added to file <top>/iocList.

apFixLinks

This command will regenerate generic links.

apFixDctRpt

This command will convert DCT short form reports to dbta report format. Executing apFixDctRpt with parm dct will force use of dct instead of atdb to read the short form report.

6.1.2 Tools invoked in the <top>/<app>App/ directory

apCreateDbDir

This command creates the database directories used by a particular application. It is executed in <top>/<app>App and is an interactive tool.

6.1.3 Tools invoked anywhere

doGets

This command uses make to ensure that all sccs controlled files in this directory and below are up–to–date.

6.2 Application Shadow Area

This section describes tools that only apply to a shadow area.

6.2.1 apCreateShadow

The first time this command is executed in a directory it creates a complete shadow area. It is also issued to fill in missing links whenever the application system area has been rebuilt.

6.2.2 apStatusSync

This command is issued whenever the shadow area must be resynced with the system area because the system area was rebuilt. It issues error messages to standard out and also writes unix commands into a file apRemoveScript. If it reports errors the user should fix the errors and reissue the apStatusSync command. When the user is satisfied then the apRemoveScript must be edited and executed. apRemoveScript contains a number of rm commands but they are commented out (preceded by #). The user should decide which files should really be removed.

6.2.3 dbsccs

A tool for managing databases in a shadow directory. See section 6.3.1 for details

6.3 Development Tools

This section describes tools that can be issued in either a system area or in a shadow area. They are issued in shadow areas during development and in the system area during integration.

6.3.1 apSccsInfo

This tool will search all directories below the current directory and list all sccs controlled files that are currently out–for–edit.

6.3.2 makesdr

The purpose of the makesdr tool is to allow application developers to build a private default.dctsdr file. Makesdr allows application developers to modify, by appending to or replacing, any EPICS ascii definition file used in creating the default.dctsdr file. The default.dctsdr file is required by DCT/GDCT. The makesdr tool also allows new ascii definition files to be introduced into the application environment.

Makesdr first searches the EPICS ascii directory followed by the cat_ascii directory and then the replace_ascii directory in order to determine the composition of each ascii definition file. The composed ascii files are then processed by cpp and the various SDR "bld" tools in order to produce SDR structures for the default.dctsdr file, record header files, and the default.sdrSum file. The end result of a successful makesdr run is that EPICS record header files etc. are either replicated or updated into a new **sdrH/rec** directory. makesdr corrects any EPICS files according to the contents of the local ascii directories. If the default.dctsdr or default.sdrSum files/links changed they are replaced with the new versions. Ascii files placed in the replace_ascii directory supercede all other ascii input files with the same name.

The application developer is expected to include the **sdrH/rec** directory when doing a vxWorks build.

Directory **sdrH/rec/** is created from scratch the first time makesdr is run. It contains either copies of EPICS header files or the versions created by makesdr.

Note: makesdr only rebuilds if something is out–of–date.

6.3.3 Buildit

This command creates a Makefile from an Imakefile. Whenever an Imakefile is modified Buildit must be executed. Note that the Imakefiles are stored in <src>, but Buildit is executed in <archV> for vxWorks and in <archU> for unix.

6.4 make

This command rebuilds various application components. What it does depends on where it is executed.

6.4.1 <top>

make doFix

runs apFixLinks

make doFixRptDct

runs apFixDctRpt dct

make doFixRpt

runs apFixDctRpt

make doGets

runs doGets

make domakesdr

runs makesdr

make bldMakefiles

Rebuids **Makefiles** from the **Imakefiles** in each application

make doapplications

Runs "make" on the makefile in each application

make doappworld

Runs "make world" on the makefile in each application

make world

Does all of the above

make

defaults to "make doapplications" above.

make dotar

Creates a compressed tar file in the directory above **<top>** and names the file **"<top>.Tar.Z".** Ex: if this was the <top> directory for par the file would be named par.Tar.Z.

make tarinfo

displays directions for unpacking the compressed tar file.

6.4.2 <shadow>

The same as 6.4.1 except that apFixLinks is a nop.

6.4.3 <app>App

See section 5.7 for what make in a application directory does.

6.4.4 <database>Db

The actions described in 5.1 are performed.

6.4.5 <archU>

The unix components are rebuilt.

6.4.6 <archV>

The IOC components are rebuilt.

APPENDIX A Application System Area

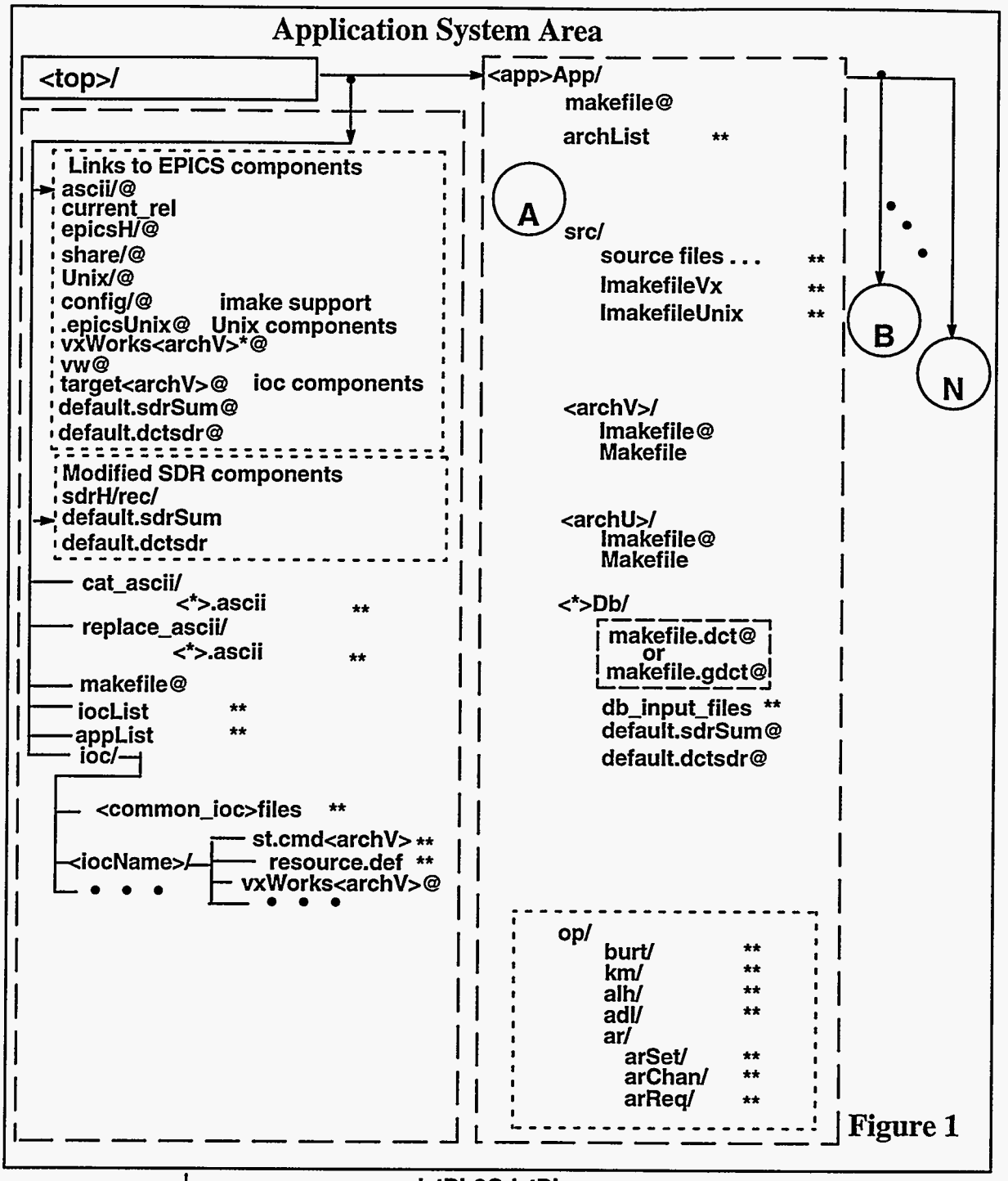

Figure 1

apCreateTop
apCreateApp
apCreateDbDir
apCreateIocName
apCreateShadow
apStatusSync
makesdr
dctDir2GdctDir
dbsccs
doGets

** user editable sccs controlled files

APPENDIX B Application Production Area Evolution

One problem that still has to be addressed is generation of a production area for use by operations. This appendix presents a possible set of procedures that could be used. Please refer to the next page for a flow diagram architecture.

1 The application manager creates and maintains the application system area (A:). This would include changing to a new EPICS release.

2 Application developers create and maintain shadow areas accessing the application system area (A:). Deltas are applied to the application system area to make changes permanent.

3 The production manager requests a new version of the application system area.

4 The application system manager fulfills this request by replicating the application system area (A:) into the application integration area (B:) and running a baseline set of regression tests.

5 The Unix system manager changes ownership of the application integration area (B:) to the application production manager.

6 The application production manager retires the (D:) previous production area.

7 The application production manager moves the current application production area (C:) to the previous production area (D:).

8 The application production manager moves the application integration area (B:) to the application production area (C:)

9 The application production manager deletes and recreates a production shadow area (E:) to be used for quick fixes.

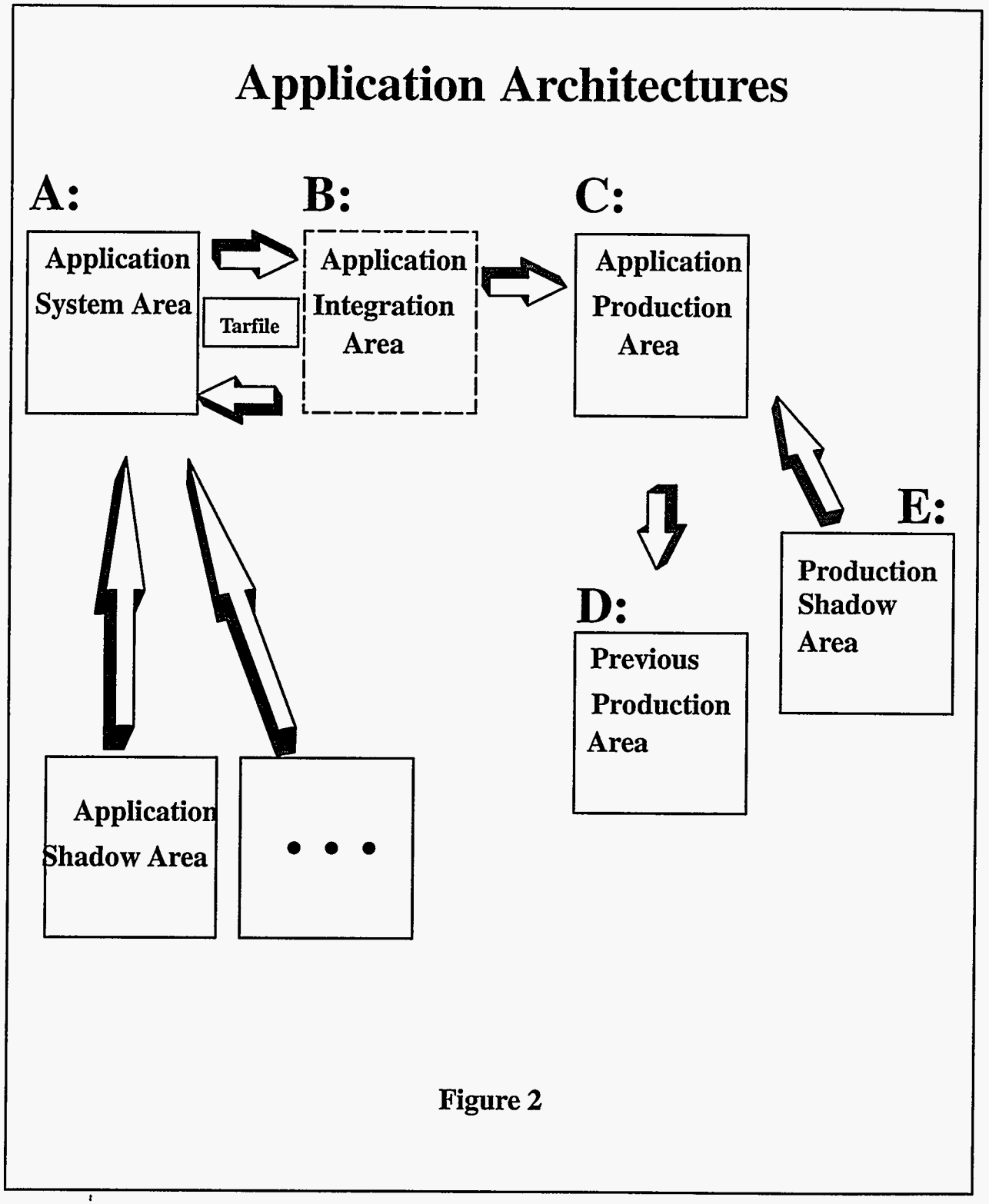
Application Architectures
A:
B:
C:
Application System Area
Tarfile
Application Integration Area
Application Production Area
E:
Production Shadow Area
D:
Previous Production Area
Application Shadow Area
• • •

Figure 2